Nama: Niansa Sixty Avilla

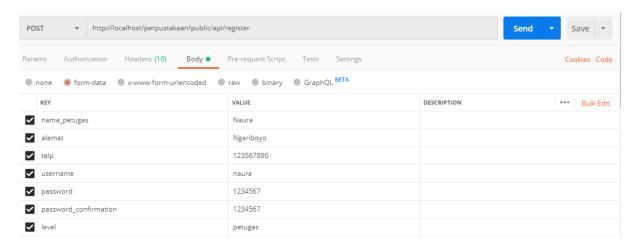
No. : 27

Kelas : XI RPL 6

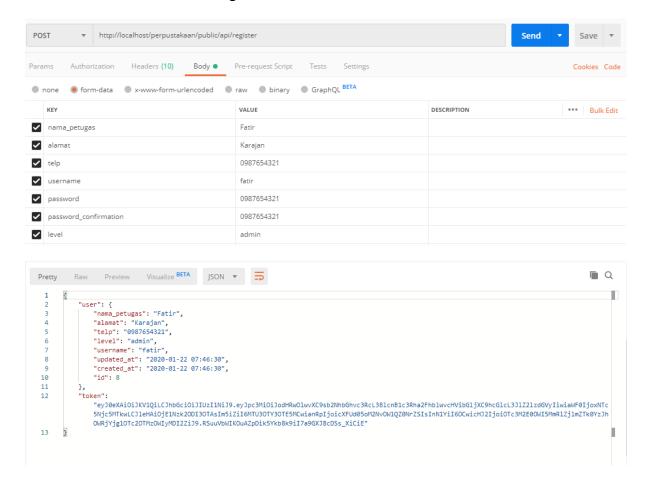
TUGAS LOGIN dan REGISTER

1. Hasil Postman

REGISTER
 Data pertama sebagai petugas

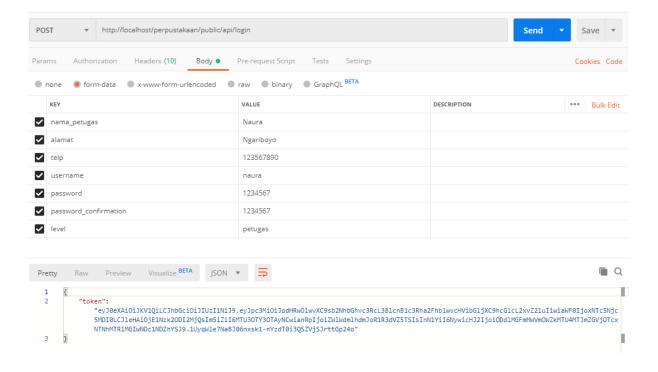


Data kedua sebagai admin

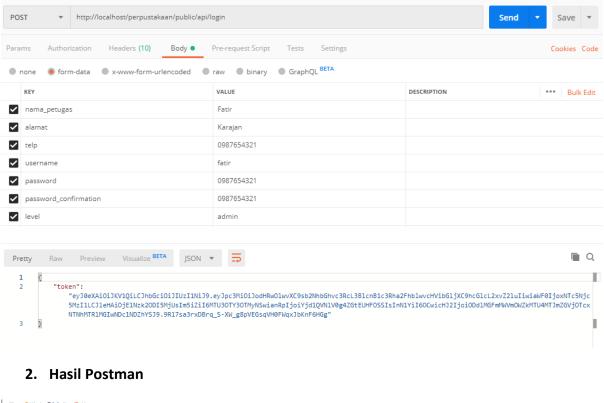


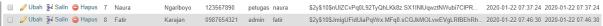
LOGIN

Data pertama sebagai petugas



Data Kedua sebagai admin





3. Controller

Petugas Controller

```
app > Http > Controllers > * PetugasController.php
  1
     <?php
  2
  3
      namespace App\Http\Controllers;
  4
  5
     use Illuminate\Http\Request;
  6
     use App\Petugas_model;
     use Illuminate\Support\Facades\Hash;
     use Illuminate\Support\Facades\Validator;
  8
 9
     use JWTAuth;
     use Tymon\JWTAuth\Exceptions\JWTException;
 10
 11
 12
     class PetugasController extends Controller
 13
 14
        public function login(Request $request)
 15
             $credentials = $request->only('username', 'password');
 16
 17
 18
              try {
                 if (! $token = JWTAuth::attempt($credentials)) {
 19
 20
                      return response()->json(['error' => 'invalid_credentials'], 400);
 21
 22
              } catch (JWTException $e) {
 23
              return response()->json(['error' => 'could_not_create_token'], 500);
 24
 25
             return response()->json(compact('token'));
 26
 27
 28
 29
          public function register(Request $request)
 30
 31
              $validator = Validator::make($request->all(), [
                  'nama_petugas' => 'required|string|max:255',
 32
 33
                  'alamat' => 'required|string|max:255',
                  'telp' => 'required|string|max:255',
 34
                  'level' => 'required',
 35
                  'username' => 'required|string|max:255',
 36
                  'password' => 'required|string|min:6|confirmed',
 37
 38
              1);
 39
```

```
38
             ]);
39
             if($validator->fails()){
40
                 return response()->json($validator->errors()->toJson(), 400);
41
42
43
44
             $user = Petugas_model::create([]
45
                'nama_petugas' => $request->get('nama_petugas'),
46
                 'alamat' => $request->get('alamat'),
                 'telp' => $request->get('telp'),
47
                 'level' => $request->get('level'),
48
49
                 'username' => $request->get('username'),
50
                  'password' => Hash::make($request->get('password')),
51
             1);
52
             $token = JWTAuth::fromUser($user);
53
54
55
             return response()->json(compact('user','token'),201);
56
57
58
         public function getAuthenticatedUser()
59
60
             try {
61
62
                 if (! $user = JWTAuth::parseToken()->authenticate()) {
63
                     return response()->json(['user_not_found'], 404);
64
65
             } catch (Tymon\JWTAuth\Exceptions\TokenExpiredException $e) {
66
67
68
                 return response()->json(['token_expired'], $e->getStatusCode());
69
             } catch (Tymon\JWTAuth\Exceptions\TokenInvalidException $e) {
70
71
72
                 return response()->json(['token_invalid'], $e->getStatusCode());
73
74
              } catch (Tymon\JWTAuth\Exceptions\JWTException $e) {
75
76
                 return response()->json(['token_absent'], $e->getStatusCode());
77
```

```
61
62
                 if (! $user = JWTAuth::parseToken()->authenticate()) {
63
                     return response()->json(['user_not_found'], 404);
64
65
             } catch (Tymon\JWTAuth\Exceptions\TokenExpiredException $e) {
66
67
                 return response()->json(['token_expired'], $e->getStatusCode());
68
69
70
             } catch (Tymon\JWTAuth\Exceptions\TokenInvalidException $e) {
71
72
                 return response()->json(['token_invalid'], $e->getStatusCode());
73
74
             } catch (Tymon\JWTAuth\Exceptions\JWTException $e) {
75
                 return response()->json(['token_absent'], $e->getStatusCode());
76
77
78
79
             return response()->json(compact('user'));
80
81
82
83
```

4. Middleware

• JwT Middleware

```
app > Http > Middleware > 🦬 JwtMiddleware.php
 1 <?php
  2
  3 namespace App\Http\Middleware;
  5 use Closure;
  6 use JWTAuth;
  7 use Exception;
  8  use Tymon\JWTAuth\Http\Middleware\BaseMiddleware;
  9
 10 class JwtMiddleware extends BaseMiddleware
 11
 12
 13
          * Handle an incoming request.
 14
 15
          * @param \Illuminate\Http\Request $request
 16
 17
          * @param \Closure $next
          * @return mixed
 18
 19
 20
         public function handle($request, Closure $next)
 21
 22
             try {
 23
                $user = JWTAuth::parseToken()->authenticate();
 24
             } catch (Exception $e) {
 25
                 if ($e instanceof \Tymon\JWTAuth\Exceptions\TokenInvalidException){
 26
                    return response()->json(['status' => 'Token is Invalid']);
                }else if ($e instanceof \Tymon\JWTAuth\Exceptions\TokenExpiredException){
 27
 28
                   return response()->json(['status' => 'Token is Expired']);
 29
                 }else{
                   return response()->json(['status' => 'Authorization Token not found']);
 30
 31
 32
 33
             return $next($request);
 34
 35
 36
```

5. Model Petugas

```
app > 🦬 Petugas_model.php
 1 <?php
  2
  3 namespace App;
  4
  5 use Illuminate\Notifications\Notifiable;
  6 use Illuminate\Contracts\Auth\MustVerifyEmail;
  7 use Illuminate\Foundation\Auth\User as Authenticatable;
    use Tymon\JWTAuth\Contracts\JWTSubject;
  8
  9
 10
     class Petugas_model extends Authenticatable implements JWTSubject
 11
 12
          use Notifiable;
          protected $table = 'petugas';
 13
 14
 15
 16
          * The attributes that are mass assignable.
 17
          * @var array
 18
 19
 20
          protected $fillable = [
          'nama_petugas', 'alamat', 'telp', 'username', 'password', 'level'
 21
 22
 23
 24
          * The attributes that should be hidden for arrays.
 25
 26
 27
          * @var array
 28
 29
          protected $hidden = [
 30
          'password', 'remember_token',
 31
          ];
 32
 33
          public function getJWTIdentifier()
 34
          return $this->getKey();
 35
 36
          public function getJWTCustomClaims()
 37
 38
 39
            return [];
 40
```

6. Routes

• Api

```
routes > 🦬 api.php
 1 <?php
  2
  3 use Illuminate\Http\Request;
  4
  5 /*
  6 /----
  7 | API Routes
  8 /----
 9
 10 | Here is where you can register API routes for your application. These
      | routes are loaded by the RouteServiceProvider within a group which
 11
      | is assigned the "api" middleware group. Enjoy building your API!
 12
 13
 14
 15
      Route::middleware('auth:api')->get('/user', function (Request $request) {
 16
 17
      return $request->user();
 18
 19 Route::post('register', 'PetugasController@register');
 20 Route::post('login', 'PetugasController@login');
 21 Route::get('/', function(){
 22
     return Auth::user()->level;
 23 })->middleware('jwt.verify');
 24
      Route::get('user', 'PetugasController@getAuthenticatedUser')->middleware('jwt.verify');
 25
 26
 27
```

7. CMD

C:\xampp\htdocs\perpustakaan>php artisan make:controller PetugasController Controller created successfully.

C:\xampp\htdocs\perpustakaan>php artisan make:model Petugas model

C:\xampp\htdocs\perpustakaan>php artisan make:middleware JwtMiddleware